Санкт-Петербургский государственный университет

Шешуков Илья Вячеславович

Выпускная квалификационная работа

Разработка вычислительных алгоритмов и программных средств поддержки принятия решений

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5004.2017 «Прикладная математика и информатика»

Профиль «Вычислительная стохастика и статистические модели»

Научный руководитель:
профессор, кафедра статистического моделирования
д. ф.-м. н., доцент Н. К. Кривулин

Рецензент:
ведущий программист,
ООО «Клауд Инструментс»
к. ф.-м. н. В. Н. Сорокин

Санкт-Петербург

2021

Saint Petersburg State University
Faculty of Mathematics and Mechanics
Department of Statistical Modelling

Sheshukov Ilia Viacheslavovich

Graduation Project

Development of computational algorithms and decision support software

Scientific Supervisor:
Professor, Department of Statistical
Modeling, D. Sc. N. K. Krivulin

Reviewer:
Lead Programmer, "Cloud Instruments" LLC,
Ph. D. V. N. Sorokin

Saint Petersburg
2021

Оглавление

Введение

Задача принятия решений на основе парных сравнений состоит в том, чтобы по результатам попарных сравнений n альтернатив определить абсолютный приоритет каждой из них. Подобные задачи возникают во множестве областей человеческой деятельности, где необходимо принимать решения на основе нескольких факторов.

Одним из методов решения подобных задач является метод log-чебышёвской аппроксимации. Преимущество этого метода в том, что решения получают в аналитическом виде, а не в численном. Однако, пока не существует общедоступных программных средств, которые бы решали данную задачу на компьютере.

В данной работе будет рассмотрен подход [8] в решении задачи оценки альтернатив на основе log-чебышёвской аппроксимации и представлено программное средство [6], решающее задачу многокритериальных парных сравнений данным методом.

Данная работа организована следующим образом: в Главе 1 и 2 рассматривается постановка задачи, в Главе 3 представлено её решение и разработанное программное средство.

Глава 1

Задача оценки альтернатив на основе парных сравнений

Задача принятия решений в самой общей формулировке состоит в следующем: из множества вариантов при заданных ограничениях нужно выбрать один или несколько вариантов, обеспечивающих решение некоторой задачи. Подобные задачи могут возникнуть в любой области, где человеку необходимо принять оптимальное решение на основе множества факторов: маркетинге, психологии, менеджменте и других.

Конкретной задачей принятия решений является задача парных сравнений. Её можно сформулировать так: имея n альтернатив и результаты сравнения каждой пары (альтернатива i «лучше» альтернативы j в m раз), необходимо упорядочить альтернативы. Также существует обобщение этой задачи на случай, когда альтернативы сравниваются по нескольким критериям, называемое *многокритериальной задачей парных сравнений*.

Обычно такие сравнения проводят люди (эксперты, респонденты опросов), поэтому результаты парных сравнений в большинстве случаев нельзя наивно упорядочить. Рассмотрим две ситуации на Рис. 1.1. В первой ситуации, упорядочить альтернативы по

($а$) Ситуация 1 ($б$) Ситуация 2

Рис. 1.1. Примеры возможных результатов парных сравнений. $T_i \to T_j$ означает «T_i лучше T_j»

возрастанию приоритета просто: T_2, T_1, T_3. Во второй же ситуации при помощи такого

наивного метода это не представляется возможным из-за цикла.

Существует множество методов решения данной задачи. Например, известны метод анализа иерархий Саати [4] и метод геометрических средних. В данной работе будет рассмотрен подход [8] в решении задачи оценки альтернатив на основе log-чебышёвской аппроксимации. Особенностью рассматриваемого подхода является то, что полученное на его основе решение может быть не единственным. Это позволяет проводить анализ решения, на основе дополнительных критериев. Также, в отличие от метода Саати, метод является аналитическим.

Целью работы является написание библиотеки и программы на языке `C++`, символьно решающих одно- и многокритериальную задачу принятия решений методом log-чебышёвской аппроксимации в max-алгебре.

1.1. Постановка задачи

Определение 1. *Матрицей парных сравнений* называется положительная матрица $\mathbf{A} = (a_{ij})$, элемент a_{ij} которой показывает, во сколько раз альтернатива i превосходит альтернативу j.

Замечание. *Для матрицы парных сравнений* $\mathbf{A} = (a_{ij})$ *выполняется*

$$a_{ij} = 1/a_{ji}, \quad i, j = 1, \ldots, n.$$

Определение 2. *Согласованной матрицей* называется матрица $\mathbf{A} = (a_{ij})$ такая, что выполняется свойство транзитивности. Т. е.

$$a_{ij} = a_{ik}a_{kj}, \quad i, j, k = 1, \ldots, n.$$

Утверждение 1. *Необходимым и достаточным условием [5] того, что матрица* $\mathbf{A}$ *согласована, является существование вектора* $\mathbf{x}$*, называемого вектором абсолютных приоритетов, такого, что*

$$a_{ij} = x_i/x_j \quad \forall i, j.$$

На практике матрицы парных сравнений обычно не согласованы, и поэтому возникает задача приближения матрицы парных сравнений согласованной матрицей. Одним из способов решения (т. н. log-чебышёвская аппроксимация [8]) этой задачи является нахождение вектора $\mathbf{x}$, который бы минимизировал следующее выражение

$$l_\infty\left(\mathbf{A}, \mathbf{x}\mathbf{x}^-\right) = \max_{1 \le i,j \le n} \left| \log a_{ij} - \log \frac{x_i}{x_j} \right|, \text{ где } \mathbf{x}^- = \begin{pmatrix} x_1^{-1} & x_2^{-1} & \ldots & x_n^{-1} \end{pmatrix},$$

Из монотонности логарифма и тождества $|x| = \max(x, -x)$ следует

$$\max_{1\le i,j\le n} \left|\log a_{ij} - \log \frac{x_i}{x_j}\right| = \log \max_{1\le i,j\le n} \max\left(\frac{a_{ij}x_j}{x_i}, \frac{x_i}{a_{ij}x_j}\right),$$

а из условия $a_{ij} = {}^1\!/_{a_{ji}}$

$$\max_{1\le i,j\le n} \max\left(\frac{a_{ij}x_j}{x_i}, \frac{x_i}{a_{ij}x_j}\right) = \max_{1\le i,j\le n} \frac{a_{ij}x_j}{x_i}.$$

Исходная задача сводится [9] к нахождению $\mathbf{x}$ такого, что

$$\min_{\mathbf{x}>0} \max_{1\le i,j\le n} \frac{a_{ij}x_j}{x_i}.$$

Полученный в результате решения задачи $\mathbf{x}$ может быть не единственным. Пусть L — множество решений. Тогда наилучшим дифференцирующим вектором называют такое решение, у которого отношение наибольшего и наименьшего элементов максимально. Аналогично, вводят и наихудший дифференцирующий вектор. Тогда наилучший дифференцирующий вектор [2] [3] находится решением задачи

$$\max_{\mathbf{x}\in L}(\max_{1\le i\le n} x_i / \min_{1\le j\le n} x_j) = \max_{\mathbf{x}\in L}(\max_{1\le i\le n} x_i \times \max_{1\le j\le n} x_j^{-1}),$$

а наихудший

$$\min_{\mathbf{x}\in L}(\max_{1\le i\le n} x_i / \min_{1\le j\le n} x_j) = \min_{\mathbf{x}\in L}(\max_{1\le i\le n} x_i \times \max_{1\le j\le n} x_j^{-1}).$$

Данная задача называется *однокритериальной*. Также существует её обобщение на случай, когда сравнение происходит не по одному критерию, а сразу по нескольким. Пусть n альтернатив сравниваются по m критериям. Матрица $\mathbf{A}_k = (a_{ij}^{(k)})$ — матрица попарных сравнений по критерию с номером $k = 1, \ldots, n$. Матрица $\mathbf{C} = (c_{ij})$ — матрица, показывающая, во сколько раз критерий с номером i важнее критерия с номером j. Необходимо построить вектор абсолютных приоритетов по матрицам $\mathbf{A}_k$ и $\mathbf{C}$ [11].

$$\min_{\mathbf{x}>0} \max_{1\le k\le m} w_k \max_{1\le i,j\le n} \frac{a_{ij}^{(k)}x_j}{x_i} = \min_{\mathbf{x}>0} \max_{1\le i,j\le n} \left(\max_{1\le k\le m} w_k a_{ij}^{(k)}\right) \frac{x_j}{x_i},$$

где коэффициенты w_k — веса — получают из матрицы $\mathbf{C}$ решая задачу

$$\min_{\boldsymbol{w}>0} \max_{1\le k,l\le m} \frac{c_{kl}w_l}{w_k}.$$

Эти задачи могут быть решены с использованием идемпотентной алгебры и max-алгебры в частности.

Как уже было сказано ранее, данный метод не гарантирует единственность решения. С одной стороны, такой результат сложнее интерпретировать; с другой, при помощи введения дополнительных объектов — *наилучшего и наихудшего дифференцирующего вектора*, можно получить дополнительную информацию о множестве решений.

1.2. Max-алгебра

Определение 3. *Max-алгебра* это алгебра над множеством $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$ с операциями $(\oplus, \times)$, где $\oplus$ — это максимум, а $\times$ — стандартное умножение [10].

1. Сложение идемпотентно
$$x \oplus x = x \quad \forall x \in \mathbb{R}_+.$$
2. Вычитание не определено.
3. Экстремальное свойство сложения
$$x \leq x \oplus y, \quad y \leq x \oplus y \quad \forall x, y \in \mathbb{R}_+.$$
4. Изотонность сложения и умножения
$$x \leq y \Rightarrow x \oplus z \leq y \oplus z, \quad xz \leq yz \quad \forall x, y, z \in \mathbb{R}_+.$$
5. Антитонность обращения
$$x \leq y \Rightarrow x^{-1} \geq y^{-1} \quad \forall x, y \in \mathbb{R}_+ \setminus \{0\}.$$
6. Эквивалентность неравенств
$$x \oplus y \leq z \Leftrightarrow x \leq z, \quad y \leq z \quad \forall x, y, z \in \mathbb{R}_+.$$
7. Степень числа вычисляется стандартным образом, как обратный элемент по умножению.

1.3. Векторная max-алгебра

Операции max-алгебры можно естественным образом расширить на векторы и матрицы.

1. Пусть $\mathbf{A} = (a_{ij})$ и $\mathbf{B} = (b_{ij})$ — матрицы размерности $m \times n$. Операции сложения и умножения на скаляр x определяются следующим образом

$$\{\mathbf{A} \oplus \mathbf{B}\}_{ij} = a_{ik} \oplus b_{kj},$$
$$\{x \times \mathbf{A}\}_{ij} = x a_{ij}.$$

2. Пусть $\mathbf{A} = (a_{ij})$ — матрица размерности $m \times n$ и $\mathbf{B} = (b_{ij})$ — матрица размерности $n \times k$. В max-алгебре их произведение $\mathbf{A} \times \mathbf{B}$ определено следующим образом

$$\{\mathbf{A} \times \mathbf{B}\}_{ij} = \bigoplus_{k=1}^{n} a_{ik} b_{kj}.$$

В дальнейшем будут использоваться понятия, определения которых приведём здесь.

Определение 4. Система векторов $(\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$ называется *линейно независимой* тогда и только тогда, когда

$$\delta(\mathbf{A}) = \min_{1 \le i \le n} \Delta\left(\mathbf{A}_{(i)}, \mathbf{a}_i\right) > 1,$$

где $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)$, $\mathbf{A}_{(i)}$ — матрица $\mathbf{A}$ без i-го столбца, $\Delta(\mathbf{A}, \mathbf{b}) \coloneqq \left(\mathbf{A}\left(\mathbf{b}^{-}\mathbf{A}\right)^{-}\right)^{-}\mathbf{b}$, при условии, что вектор $\mathbf{b}$ не имеет нулевых координат — функция невязки.

Определение 5. Пусть $\mathbf{A} = (a_{ij})$ — матрица размерности $n \times n$. *Следом* матрицы $\mathbf{A}$ называется величина

$$\operatorname{tr} \mathbf{A} \coloneqq \bigoplus_{i=1}^{n} a_{ii}.$$

Определение 6. *Тропический определитель* матрицы это величина

$$\operatorname{tr} \mathbf{A} \oplus \cdots \oplus \operatorname{tr} \mathbf{A}^n.$$

Тропический определитель матрицы $\mathbf{A}$ обозначается как $\operatorname{Tr} \mathbf{A}$.

Определение 7. *Спектральным радиусом* матрицы $\mathbf{A}$ называется её максимальное собственное число λ. Он вычисляется по формуле

$$\lambda = \operatorname{tr} \mathbf{A} \oplus \cdots \oplus \operatorname{tr}^{1/n} \mathbf{A}^n.$$

Определение 8. Оператор Клини

Если $\operatorname{Tr}\mathbf{A} \leq 1$, то *оператор Клини* определяется следующим образом

$$\mathbf{A}^* \coloneqq \mathbf{I} \oplus \mathbf{A} \oplus \cdots \oplus \mathbf{A}^{n-1}.$$

1.4. Задачи тропической оптимизации

Здесь приведём задачи оптимизации, которые будут использоваться далее, и их решения [2]. Задача

$$\min_{\mathbf{x}>0} \mathbf{x}^{-}\mathbf{A}\mathbf{x}$$

имеет решение вида

$$\mathbf{x} = (\lambda^{-1}\mathbf{A})^*\mathbf{u},$$

где λ — спектральный радиус матрицы $\mathbf{A}$, $\mathbf{u}$ — произвольный положительный вектор. Задача нахождения наихудшего дифференцирующего вектора

$$\min_{\mathbf{x}\in L} \mathbf{1}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{-}\mathbf{1}$$

имеет решение вида

$$\mathbf{x} = (\delta^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus \lambda^{-1}\mathbf{A})^*\mathbf{u},$$

где λ — спектральный радиус матрицы $\mathbf{A}$, $\delta = \mathbf{1}^{\mathrm{T}}(\lambda^{-1}A)^*\mathbf{1}$, $\mathbf{u}$ — произвольный положительный вектор. Задача нахождения наилучшего дифференцирующего вектора

$$\max_{\mathbf{x}\in L} \mathbf{1}^{\mathrm{T}}\mathbf{x}\mathbf{x}^{-}\mathbf{1}$$

имеет решение вида

$$\mathbf{x} = \mathbf{B}\left(\mathbf{I} \oplus \mathbf{B}_{lk}^{-}\mathbf{B}\right)\mathbf{u},$$

где $\mathbf{B} = (\lambda^{-1}A)^*$, $\mathbf{b}_j$ — j-й столбец матрицы $\mathbf{B}$, матрица $\mathbf{B}_{lk}$ получена из $\mathbf{B}$ обнулением всех элементов, кроме b_{lk}, а индексы k и l находят из условия

$$\mathbf{1}^T\boldsymbol{b}_k b_{lk}^{-1} = \Delta.$$

Глава 2

Многокритериальная задача парных сравнений

2.1. Постановка многокритериальной задачи

В терминах max-алгебры задача принимает вид

$$\min_{\mathbf{x}>\mathbf{0}} \mathbf{x}^{-}\mathbf{A}\mathbf{x} \quad \text{при} \quad \mathbf{A} = \bigoplus_{1\le k\le m} w_k \mathbf{A}_k,$$

где вектор весов $\mathbf{w} = (w_k)$ находится как решение задачи

$$\min_{\mathbf{w}>0} \mathbf{w}^{-}\mathbf{C}\mathbf{w}.$$

Все решения задачи определения весов записываются в виде

$$\mathbf{w} = (\lambda^{-1}\mathbf{C})^{*}\mathbf{v},\ \mathbf{v} > 0,$$

где λ — спектральный радиус матрицы $\mathbf{C}$. Аналогично, если μ — спектральный радиус матрицы $\mathbf{A}$, то все решения задачи оценки рейтингов альтернатив имеют вид:

$$\mathbf{x} = (\mu^{-1}\mathbf{A})^{*}\mathbf{u},\ \mathbf{u} > 0.$$

2.2. План решения многокритериальной задачи

1. По матрице $\mathbf{C}$ определяем вектор весов критериев:

$$\mathbf{w} = (\mu^{-1}\mathbf{C})^{*}\mathbf{u}, \quad \mathbf{u} > 0, \quad \mu = \mathrm{tr}(\mathbf{C_1}) \oplus \ldots \oplus \mathrm{tr}^{1/m}(\mathbf{C_m}).$$

2. Если полученный вектор $\mathbf{w}$ не единственный, то находятся наихудший и наилучший дифференцирующий вектор весов:

$$\mathbf{w}_1 = (\delta^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus \mu^{-1}\mathbf{C})^{*}\mathbf{v_1}, \quad \mathbf{v_1} > 0, \quad \delta = \mathbf{1}^{\mathrm{T}}(\mu^{-1}C)^{*}\mathbf{1},$$

и

$$\mathbf{w}_2 = \mathbf{P}(\mathbf{I} \oplus \mathbf{P}_{lk}^{-}\mathbf{P})\mathbf{v_2}, \quad \mathbf{v_2} > 0,$$

для которого матрица $\mathbf{P}$ получена из матрицы $(\mu^{-1}\mathbf{C})^{*}$ вычеркиванием линейно-зависимых столбцов, матрица $\mathbf{P}_{lk}$ — из матрицы $\mathbf{P} = (p_{ij})$ заменой на ноль всех элементов, кроме p_{lk}, а индексы k и l определяются, исходя из условий

$$k = \arg\max_j \mathbf{1}^{\mathrm{T}}\mathbf{p}_j\mathbf{p}_j^{-}\mathbf{1}, \quad l = \arg\max_i p_{ik}^{-1}.$$

3. С помощью векторов $\mathbf{w}_1 = (w_i^{(1)})$ и $\mathbf{w}_2 = (w_i^{(2)})$ составляются взвешенные суммы матриц парных сравнений:

$$\mathbf{D}_1 = w_1^{(1)}\mathbf{A_1} \oplus \ldots \oplus w_m^{(1)}\mathbf{A_m},$$
$$\mathbf{D}_2 = w_1^{(2)}\mathbf{A_1} \oplus \ldots \oplus w_m^{(2)}\mathbf{A_m}.$$

4. Вычисляется вектор рейтингов альтернатив для матрицы $\mathbf{D_1}$

$$\mathbf{x} = (v_1^{-1}\mathbf{D_1})^*\mathbf{u}_1, \quad \mathbf{u}_1 > 0, \quad v_1 = \operatorname{tr}\mathbf{D}_1 \oplus \cdots \oplus \operatorname{tr}^{1/n}\mathbf{D}_1^n.$$

5. Если полученный вектор не единственный, то вместо него ищется наихудший дифференцирующий вектор

$$\mathbf{x}_1 = (\delta_1^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus v_1^{-1}\mathbf{D}_1)^*\mathbf{u_1}, \quad \mathbf{u}_1 > 0, \quad \delta_1 = \mathbf{1}^{\mathrm{T}}(v^{-1}\mathbf{D_1})^*\mathbf{1}.$$

6. Вычисляется вектор рейтингов альтернатив для матрицы $\mathbf{D_2}$

$$\mathbf{x}_2 = (v_2^{-1}\mathbf{D}_2)^*\mathbf{u}_2, \quad \mathbf{u_2} > 0, \quad v_2 = \operatorname{tr}\mathbf{D}_2 \oplus \ldots \oplus \operatorname{tr}^{1/n}(\mathbf{D}_2^n).$$

7. Если этот вектор не единственный, то вместо него рассматриваем наилучший дифференцирующий вектор

$$\mathbf{x}_2 = \mathbf{S}(\mathbf{I} \oplus \mathbf{S}_{lk}^{-}\mathbf{S})\mathbf{u}_2, \ \mathbf{u}_2 > 0,$$

где матрица $\mathbf{S}$ получена из матрицы $(v_2^{-1}\mathbf{D}_2)^*$ вычеркиванием линейно-зависимых столбцов, матрица $\mathbf{S}_{lk}^{-}$ — из матрицы $S = (s_{ij})$ обращением в нуль всех элементов, кроме s_{lk}, а индексы k и l определяются, исходя из условий

$$k = \arg\max_j \ \mathbf{1}^{\mathrm{T}}\mathbf{s}_j\mathbf{s}_j^{-}\mathbf{1}, \ l = \arg\max_i \ s_{ik}^{-}.$$

Глава 3

Разработка программных средств

Разработанное программное средство представляет собой расширение для библиотеки линейной алгебры `Eigen` [1] (версия 3.3.8), позволяющее работать с символьными вычислениями в max-алгебре, а так же реализацию ряда функций, используемых в алгоритме решения задачи (таких как нахождение тропического определителя, спектрального радиуса, оператора Клини, нахождение линейно независимых векторов и других), и непосредственно сам алгоритм. Для работы с символьными вычислениями использовалась библиотека `GiNaC` [7] (версия 1.8.0).

3.1. Пример решения многокритериальной задачи

3.1.1. Условие

$$\mathbf{C} = \begin{pmatrix} 1 & 1/3 \\ 3 & 1 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} 1 & 3 & 1/3 \\ 1/3 & 1 & 1 \\ 3 & 1 & 1 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 1/3 & 5 \\ 3 & 1 & 7 \\ 1/5 & 1/7 & 1 \end{pmatrix}.$$

3.1.2. Решение

Найдём спектральный радиус матрицы $\mathbf{C}$:

$$\lambda = \bigoplus_{k=1}^{2} \operatorname{tr}^{1/k}\left(\mathbf{C}^k\right) = \operatorname{tr}\mathbf{C} \oplus \sqrt{\operatorname{tr}\mathbf{C}^2} = \operatorname{tr}\begin{pmatrix} 1 & 1/3 \\ 3 & 1 \end{pmatrix} \oplus \sqrt{\operatorname{tr}\begin{pmatrix} 1 & 1/3 \\ 3 & 1 \end{pmatrix}} = 1.$$

В нашем случае, матрица Клини равна самой матрице $\mathbf{C}$.

$$\mathbf{C}^* = \mathbf{I} \oplus \mathbf{C} = \mathbf{C}.$$

Тогда

$$(\lambda^{-1}\mathbf{C})^*\mathbf{v} = (\lambda^{-1})\mathbf{C}\mathbf{v} = \mathbf{C}\mathbf{v}, \quad \mathbf{v} > 0.$$

Столбцы матрицы коллинеарны, поэтому просто возьмём первый из них. Вектор весов критериев

$$\mathbf{w} = \begin{pmatrix} 1 & 3 \end{pmatrix}.$$

Вычислим матрицу $\mathbf{B}$:

$$\mathbf{B} = \bigoplus_{k=1}^{2} w_k \mathbf{A}_k = \begin{pmatrix} 1 & 3 & 1/3 \\ 1/3 & 1 & 1 \\ 3 & 1 & 1 \end{pmatrix} \oplus 3 \begin{pmatrix} 1 & 1/3 & 5 \\ 3 & 1 & 7 \\ 1/5 & 1/7 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 3 & 15 \\ 9 & 3 & 21 \\ 3 & 1 & 3 \end{pmatrix}.$$

Спектральный радиус этой матрицы равен $\mu = \sqrt{45}$. Тогда вектор рейтингов альтернатив:

$$\mathbf{x} = \mathbf{S}\mathbf{u} = \left(\frac{1}{\sqrt{45}} B\right)^{*} \mathbf{u} = \mathbf{I} \oplus \frac{1}{\sqrt{45}}\mathbf{B} \oplus (\frac{1}{\sqrt{45}}\mathbf{B})^2 =$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \oplus \begin{pmatrix} 1/\sqrt{5} & 1/\sqrt{5} & \sqrt{5} \\ 3/\sqrt{5} & 1/\sqrt{5} & 7/\sqrt{5} \\ 1/\sqrt{5} & 1/(3\sqrt{5}) & 1/\sqrt{5} \end{pmatrix} \oplus$$

$$\oplus \begin{pmatrix} 1 & 1/3 & 7/5 \\ 7/5 & 3/5 & 3 \\ 1/5 & 1/5 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1/\sqrt{5} & \sqrt{5} \\ 7/5 & 1 & 7/\sqrt{5} \\ 1/\sqrt{5} & 1/5 & 1 \end{pmatrix} \mathbf{u}.$$

Здесь только первый и второй столбцы являются линейно независимыми, значит будем искать наихудший и наилучший дифференцирующий вектор.

Наихудший дифференцирующий вектор

$$\delta = \mathbf{1}^{\mathrm{T}} (\mu^{-1}\mathbf{B})^{*}\mathbf{1} = 7/\sqrt{5}.$$

$$\mathbf{x_2} = (\delta^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus \mu^{-1}\mathbf{B})^{*}\mathbf{u} = \begin{pmatrix} \sqrt{45}/15 & \sqrt{45}/15 & \sqrt{45}/3 \\ \sqrt{45}/5 & \sqrt{45}/15 & 7\sqrt{45}/15 \\ \sqrt{45}/15 & \sqrt{45}/45 & \sqrt{45}/15 \end{pmatrix}^{*} u =$$

$$= \mathbf{I} \oplus (\delta^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus \mu^{-1}\mathbf{B}) \oplus (\delta^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}} \oplus \mu^{-1}\mathbf{B})^2 =$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \oplus \begin{pmatrix} 1/\sqrt{5} & 1/\sqrt{5} & \sqrt{5} \\ 3/\sqrt{5} & 1/\sqrt{5} & 7/\sqrt{5} \\ 1/\sqrt{5} & 3\sqrt{5}/7 & 1/\sqrt{5} \end{pmatrix} \oplus \begin{pmatrix} 1 & 5/7 & 7/5 \\ 7/5 & 1 & 3 \\ 3/7 & 1/5 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 5/7 & \sqrt{5} \\ 7/5 & 1 & 7/\sqrt{5} \\ 1/\sqrt{5} & \sqrt{5}/7 & 1 \end{pmatrix} \mathbf{u}, \quad \mathbf{u} > 0.$$

Тут все столбцы коллинеарны. Возьмём первый и нормируем:

$$\begin{pmatrix} 0.3512 \\ 0.4917 \\ 0.1570 \end{pmatrix}.$$

Порядок альтернатив: (II) $\succ$ (I) $\succ$ (III).

Наилучший дифференцирующий вектор

Найденный нами наихудший дифференцирующий вектор, это первый столбец матрицы $\mathbf{S}$. Можем предположить, что наилучший дифференцирующий вектор окажется вторым столбцом этой матрицы. Давайте это проверим.

Из предыдущего получили, что $\mathbf{S} = \begin{pmatrix} 1 & 1/\sqrt{5} \\ 7/5 & 1 \\ 1/\sqrt{5} & 1/5 \end{pmatrix}$. Для нахождения наилучшего дифференцирующего вектора найдём число Δ:

$$\Delta = \mathbf{1}^{\mathrm{T}}\mathbf{S}\mathbf{S}^{-}\mathbf{1} = \begin{pmatrix} 7/5 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{5} \\ 5 \end{pmatrix} = 5.$$

Все наилучшие дифференцирующие решения имеют вид

$$\mathbf{x}_1 = \mathbf{S}\left(\mathbf{I} \oplus \mathbf{S}_{lk}^{-}\mathbf{S}\right)\mathbf{v}, \quad \mathbf{v} > \mathbf{0},$$

где индексы k и l определяются из условия

$$\mathbf{1}^{\mathrm{T}}\mathbf{s}_k s_{lk}^{-1} = \Delta = 5.$$

Это условие выполняется только при $k = 2$ и $l = 3$. Тогда наилучшие решения генерируют столбцы матрицы:

$$\mathbf{x}_1 = \mathbf{S}\left(\mathbf{I} \oplus \mathbf{S}_{3,2}^{-}\mathbf{S}\right)\mathbf{v} =$$
$$= \begin{pmatrix} 1 & 1/\sqrt{5} \\ 7/5 & 1 \\ 1/\sqrt{5} & 1/5 \end{pmatrix} \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \oplus \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} 1 & 1/\sqrt{5} \\ 7/5 & 1 \\ 1/\sqrt{5} & 1/5 \end{pmatrix} \right) \mathbf{v} =$$
$$= \begin{pmatrix} 1 & 1/\sqrt{5} \\ \sqrt{5} & 1 \\ 1/\sqrt{5} & 1/5 \end{pmatrix} \mathbf{v}.$$

Столбцы этой матрицы коллинеарны, возьмём второй.

Как и предполагалось, наилучший дифференцирующий вектор коллинеарен второму столбцу матрицы $\mathbf{S}$.

$$x_1 = \begin{pmatrix} 1/\sqrt{5} \\ 1 \\ 1/5 \end{pmatrix} \approx \begin{pmatrix} 0.4472 \\ 1 \\ 0.2 \end{pmatrix}.$$

Порядок альтернатив: (II) $\succ$ (I) $\succ$ (III).

Общий ответ: (II) $\succ$ (I) $\succ$ (III).

3.1.3. Использование библиотеки для вычислений

Приведём пример вывода программы для получения вышеприведённых вычислений на компьютере (более сложный и подробный пример может быть найден в Приложении А.3).

```
Computing spectral radius: lambda = 1


Computing (1/lambda * C)^*
(1/lambda * C)^* =
[1, 1/3]
 [3, 1]


Computing delta:
delta = 3, delta^-1 = 1/3
```

```
Computing the worst differentiating weight vector
Linearly independent w1 =
[1/3]
 [1]

Worst differentiating weight vector =
[1/3]
 [1]

D1 = [1, 1, 5]
 [3, 1, 7]
 [1, 1/3, 1]
nu_1 = sqrt(5)

Worst differentiating vector of alternatives
x1 =
[sqrt(5)]
 [7/5*sqrt(5)]
 [1]



Computing the best differenting weight vector
P =
[1/3]
 [1]

Computing w_2

Linearly independent w2 =
[1/3]
 [1]
```

```
Computing the best differentiating vector of alternatives
D2 = [1, 1, 5]
 [3, 1, 7]
 [1, 1/3, 1]
nu_2 = sqrt(5)

Best differentiang vector of alternatives
x2 = [sqrt(5)]
 [5]
 [1]
```

Получили следующий порядок альтернатив: для наихудшего дифференцирующего вектора (II) $\succ$ (I) $\succ$ (III), для наилучшего — (II) $\succ$ (I) $\succ$ (III).

Результат совпадает с результатом, полученным ручными вычислениями.

3.1.4. Внутреннее устройство библиотеки

Библиотека представляет собой

1. Класс `MaxAlgebra<T>` реализующий операции в max-алгебре.

```
Template<typename Op>
class MaxAlgebra {
 public:
  GiNaC::ex value;
  MaxAlgebra() : value(){};
  ...
  friend MaxAlgebra operator+(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend MaxAlgebra operator*(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend MaxAlgebra operator/(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend bool operator<(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend bool operator==(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend bool operator>(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend MaxAlgebra pow(const MaxAlgebra& lhs, const MaxAlgebra& rhs);
  friend std::ostream& operator<<(std::ostream& out, const MaxAlgebra& val);
  ...
  MaxAlgebra& operator=(const MaxAlgebra& rhs);
```

```
  MaxAlgebra& operator+=(const MaxAlgebra& rhs);
  MaxAlgebra& operator*=(const MaxAlgebra& rhs);
  MaxAlgebra& operator/=(const MaxAlgebra& rhs);
  MaxAlgebra& abs(const MaxAlgebra& rhs);
};
```

2. Специализация класса реализующая операции в max-алгебре.

```
using MaxTimes = MaxAlgebra<std::multiplies<void>>;
using MaxPlus = MaxAlgebra<std::plus<void>>;

MaxPlus operator*(const MaxPlus& lhs, const MaxPlus& rhs);
MaxPlus operator/(const MaxPlus& lhs, const MaxPlus& rhs);
...
MaxTimes operator*(const MaxTimes& lhs, const MaxTimes& rhs);
MaxTimes operator/(const MaxTimes& lhs, const MaxTimes& rhs);
```

3. Расширение для библиотеки `Eigen`, позволяющее использовать `MaxAlgebra<T>` в качестве элементов матрицы.

4. Набор функций, участвующих в алгоритме решения задачи: нахождение тропического определителя, спектрального радиуса, оператор Клини, нахождение линейно независимых векторов и т. д.

3.1.5. Список реализованных функций

1. Арифметические операции над числами в max-алгебре: $\oplus$, $\times$, возведение в степень и т. д.

2. Операции над векторами и матрицами: перемножение, транспонирование, возведение в степень и т. д.

3. Функции, необходимые в решении задачи: спектральный радиус, получение матрицы Клини, нахождение коэффициентов в задаче нахождение наилучшего дифференцирующего вектора.

Заключение

В ходе данной работы были изучены различные методы решения задач оценки альтернатив на основе парных сравнений в одно- и многокритериальной постановке. Был исследован подход, основанный на log-чебышёвской аппроксимации матриц парных сравнений. Также, были изучены методы тропической (идемпотентной) математики решения задач оценки альтернатив на основе парных сравнений.

На языке `C++` с применением библиотек `Eigen` и `GiNaC` было разработано программной средство (библиотека и программа) [6], реализующее как аппарат тропической математики, так и метод log-чебышёвской аппроксимации. Программа распространяется под свободной лицензией.

Разработанное средство обеспечивает возможность полного решения задачи в полностью автоматическом режиме.

Были проведены эксперименты, тестирующие работу программы в решении нескольких многокритериальных задач попарных сравнений. Результаты, полученные в результате работы программы совпадают с результатами, полученными ручным подсчётом по данному методу.

Список литературы

1. Eigen v3 / G. Guennebaud, B. Jacob [и др.]. — 2010. — http://eigen.tuxfamily.org.
2. Krivulin N. Methods of Tropical Optimization in Rating Alternatives Based on Pairwise Comparisons // Operations Research Proceedings 2016 / под ред. A. Fink, A. Fügenschuh, M. J. Geiger. — Cham : Springer International Publishing, 2018. — с. 85—91. — (Operations Research Proceedings). — ISBN 978-3-319-55702-1. — DOI: 10.1007/978-3-319-55702-1_13.
3. Krivulin N., Sergeev S. Tropical implementation of the Analytical Hierarchy Process decision method // Fuzzy Sets and Systems. — 2019. — дек. — т. 377. — с. 31—51. — ISSN 01650114. — DOI: 10.1016/j.fss.2018.10.013. — URL: https://linkinghub.elsevier.com/retrieve/pii/S0165011418308145 (дата обр. 27.05.2021).
4. Saaty T. L. The analytic hierarchy process: planning, priority setting, resource allocation. — New York ; London : McGraw-Hill International Book Co, 1980. — 287 с. — ISBN 978-0-07-054371-3.
5. Saaty T. L., Vargas L. G. Comparison of eigenvalue, logarithmic least squares and least squares methods in estimating ratios // Mathematical Modelling. — 1984. — с. 309—324.
6. Sheshukov I. isheshukov/vkr-2021: A software for multiple- criteria decision-making. — вер. v1.0. — 05.2021. — DOI: 10.5281/zenodo.4793450. — URL: https://zenodo.org/record/4793450.
7. Vollinga J. GiNaC - Symbolic computation with C++ // Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. — 2006. — апр. — т. 559, № 1. — с. 282—284. — ISSN 01689002. — DOI: 10.1016/j.nima.2005.11.155. — arXiv: hep-ph/0510057. — URL: http://arxiv.org/abs/hep-ph/0510057 (дата обр. 21.05.2021).
8. Кривулин Н. К., Агеев В. А. Методы тропической оптимизации в многокритериальных задачах оценки альтернатив на основе парных сравнений // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2019. — дек. — с. 472—488.

9. Кривулин Н. К., Агеев В. А., Гладких И. В. Применение методов тропической оптимизации для оценки альтернатив на основе парных сравнений // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. — 2017. — с. 27—41.

10. Кривулин Н. К. Методы идемпотентной алгебры в задачах моделирования и анализа сложных систем. — Издательство Санкт-Петербургского государственного университета, 29.07.2009. — 258 с. — ISBN 978-5-288-04906-4. — Google-Books-ID: PDQP7kIGrhMC.

11. О решении многокритериальных задач принятия решений на основе парных сравнений / Н. К. Кривулин [и др.] // Компьютерные инструменты в образовании. — 2020. — № 2. — с. 27—58. — ISSN 2071-2359. — DOI: 10.32603/2071-2340-2020-2-27-58. — URL: http://cte.eltech.ru/ojs/index.php/kio/article/view/1655.

Приложение А

Примеры работы программы

А.1. Пример 1

Объявление двух матриц A_1 и A_2 размера 3×3:

```
MatrixDn A1(3, 3);
MatrixDn A2(3, 3);

A1 << ex(sin(1)), 3, ex(1) / 3,
      ex(1) / 3, 1, 1,
                3, 1, 1;
A2 <<   0, ex(1) / 3, 5,
        3,          1, 7,
ex(1) / 5, ex(1) / 7, 1;
```

Сумма и произведение этих матриц:

```
std::cout << "A1+A2=\n" << A1+A2 << std::endl;
std::cout << "A1*A2=\n" << A1*A2 << std::endl;
```

Вывод:

```
A1 + A2 =                A1 * A2 =
  \sin(1) 3 5              9  3 21
  3       1 7              3  1  7
  3       1 1              3  1 15
```

А.2. Пример 2

Объявление матрицы C размера 4×4:

```
MatrixDn C(4, 4);
C <<  8, 3, 10, 6,
      7, 2,  1, 4,
     10, 8,  2, 2,
```

```
        4, 7,  6, 1;
```

Вычисление спектрального радиуса:

```
   std::cout
         << "Spectral␣radius␣of␣C␣=␣"
         << spectral_radius(C)
         << std::endl;
```

Вывод:

```
m^1=                        m^2=
 8   3 10   6               100   80   80   48
 7   2   1   4                56   28   70   42
10   8   2   2                80   30 100   60
 4   7   6   1                60   48   40   28
                            tr(m^2)^(1/2) = 10

m^3=                        m^4=
 800   640 1000   600        10000   8000   8000   4800
 700   560   560   336         5600   4480   7000   4200
1000   800   800   480         8000   6400 10000   6000
 480   320   600   360         6000   4800   4800   2880
tr(m^3)^(1/3) = 800^(1/3)    tr(m^4)^(1/4) = 10

Spectral radius of C = 10
```

A.3. Пример 3. Решение многокритериальной задачи парного сравнения

Условие

$$\mathbf{C} = \begin{pmatrix} 1 & 1/4 & 1/5 & 1/4 & 6 & 1/6 \\ 4 & 1 & 1/3 & 3 & 6 & 1/2 \\ 5 & 3 & 1 & 4 & 7 & 3 \\ 4 & 1/3 & 1/4 & 1 & 5 & 1/5 \\ 1/6 & 1/6 & 1/7 & 1/5 & 1 & 1/7 \\ 6 & 2 & 1/3 & 5 & 7 & 1 \end{pmatrix}$$

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 5 & 8 \\ 1/5 & 1 & 5 \\ 1/8 & 1/5 & 1 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 7 & 9 \\ 1/7 & 1 & 7 \\ 1/9 & 1/7 & 1 \end{pmatrix}, \quad \mathbf{A}_3 = \begin{pmatrix} 1 & 1/7 & 1/9 \\ 7 & 1 & 1/7 \\ 9 & 7 & 1 \end{pmatrix},$$

$$\mathbf{A}_4 = \begin{pmatrix} 1 & 3 & 5 \\ 1/3 & 1 & 4 \\ 1/5 & 1/4 & 1 \end{pmatrix}, \quad \mathbf{A}_5 = \begin{pmatrix} 1 & 3 & 5 \\ 1/3 & 1 & 4 \\ 1/5 & 1/4 & 1 \end{pmatrix}, \quad \mathbf{A}_6 = \begin{pmatrix} 1 & 7 & 9 \\ 1/7 & 1 & 7 \\ 1/9 & 1/7 & 1 \end{pmatrix}$$

Код программы

Задание матриц:

```
MatrixDn C(6, 6);
C << 1, MaxTimes(1, 4), MaxTimes(1, 5), MaxTimes(1, 4), 6,
    MaxTimes(1, 6), 4, 1, MaxTimes(1, 3), 3, 6, MaxTimes(1, 2), 5, 3, 1,
    4, 7, 3, 4, MaxTimes(1, 3), MaxTimes(1, 4), 1, 5, MaxTimes(1, 5),
    MaxTimes(1, 6), MaxTimes(1, 6), MaxTimes(1, 7), MaxTimes(1, 5), 1,
    MaxTimes(1, 7), 6, 2, MaxTimes(1, 3), 5, 7, 1;


std::vector<MatrixDn>As;
MatrixDn A1(3, 3);
A1 << 1, 5, 8, ma(1, 5), 1, 5, ma(1, 8), ma(1, 5), 1;
MatrixDn A2(3, 3);
A2 << 1, 7, 9, ma(1, 7), 1, 7, ma(1, 9), ma(1, 7), 1;
MatrixDn A3(3, 3);
```

```
A3 << 1, ma(1, 7), ma(1, 9), 7, 1, ma(1, 7), 9, 7, 1;
MatrixDn A4(3, 3);
A4 << 1, 3, 5, ma(1, 3), 1, 4, ma(1, 5), ma(1, 4), 1;
MatrixDn A5(3, 3);
A5 << 1, 3, 5, ma(1, 3), 1, 4, ma(1, 5), ma(1, 4), 1;
MatrixDn A6(3, 3);
A6 << 1, 7, 9, ma(1, 7), 1, 7, ma(1, 9), ma(1, 7), 1;

As.push_back(A1);
As.push_back(A2);
As.push_back(A3);
As.push_back(A4);
As.push_back(A5);
As.push_back(A6);

const auto num_crit_mat = C.rows();
const auto num_comp_mat = A1.rows();
```

Вычисление спектрального радиуса:

```
auto lambda = spectral_radius(C);
```

Вычисление $(\lambda^{-1}C)^*$:

```
MatrixDn Calmoststar = (1 / lambda) * C;
auto w = cleany(Calmoststar);
```

Вычисление δ:

```
auto delta = (MatrixDn::Ones(1, num_crit_mat) * w * MatrixDn::Ones(num_crit_mat, 1
```

Вычисление наихудшего дифференцирующего вектора:

```
MatrixDn W1m = (1 / delta * MatrixDn::Ones(num_crit_mat, num_crit_mat)) + Calmosts
auto w1 = cleany(W1m);
auto liw1 = linearly_independent_system(w1);
```

Получилась матрица из двух линейно независимых векторов. Попробуем получить «самый наихудший» вектор. Возьмём линейнонезависимые столбцы матрицы, нормируем по максимуму и сложим:

```
MatrixDn worst_vector = MatrixDn::Zero(liw1.rows(), 1);
```

```
for (int i = 0; i < liw1.cols(); ++i){
    worst_vector += liw1.col(i) / liw1.col(i).maxCoeff();
}
```

Получим взвешенную сумму:

```
MatrixDn D1 = MatrixDn::Zero(num_comp_mat, num_comp_mat);
for (int i = 0; i < worst_vector.rows(); ++i){
    D1 += worst_vector(i) * As[i];
}

auto nu_1 = spectral_radius(D1);
MatrixDn almost_x1 = (1 / nu_1) * D1;
auto x1 = cleany(almost_x1);
auto LI_x1 = linearly_independent_system(x1);
```

Вычисляем вектор рейтингов альтернатив матрицы $\mathbf{D}_1$. Если матрица в переменной `LI_x1` состоит из одного столбца, то этот столбец является решением. В противном случае неединственности решения:

```
auto delta_1 = (MatrixDn::Ones(1, x1.rows()) *
  x1 *
  MatrixDn::Ones(x1.cols(), 1)).value();

MatrixDn almost_x1new = ((1/delta_1) *
  MatrixDn::Ones(x1.cols(), x1.rows())) +
  almost_x1;

auto x1new = cleany(almost_x1new);

auto LI_x1new = linearly_independent_system(x1new);
```

Вычисление наилучшего дифференцирующего вектора. Матрица P — линейнонезависимые столбцы матрицы w.

```
MatrixDn P = linearly_independent_system(w);
```

Получение коэффициентов k и l:

```
auto bdfc = best_diff_vector_coefficients(P);
```

Вычисление w_2. На этом этапе вычислим w_2 для каждой пары коэфициентов из предыдущего шага, объединим в одну матрицу и найдём линейнонезависимую часть.

```
for (auto& [k, l]: bdfc){

    MatrixDn Plk_inverse = MatrixDn::Zero(P.rows(), P.cols());
    Plk_inverse(l, k) = 1 / P(l, k);
    auto t = P * (MatrixDn::Identity(P.cols(), P.cols()) +
                  Plk_inverse.transpose() * P);
    W2.conservativeResize(t.rows(), W2.cols() + t.cols());
    W2.rightCols(t.cols()) = t;
}
auto LI_W2 = linearly_independent_system(W2);
auto nu_2 = spectral_radius(D2);
MatrixDn almost_x2 = (1 / nu_2) * D2;

auto x2 = cleany(almost_x2);
auto LI_x2 = linearly_independent_system(x2);
```

Вычислияем вектор рейтингов альтернатив для $\mathbf{D}_2$. Если матрица в переменной `LI_x2` состоит из одного столбца, то этот столбец является решением. В противном случае неединственности решения:

```
auto S = LI_x2;
auto bdfc = best_diff_vector_coefficients(S);
MatrixDn WS;
for (auto& [k, l]: bdfc){
    MatrixDn Slk_inverse = MatrixDn::Zero(S.rows(), S.cols());
    Slk_inverse(l, k) = 1 / S(l, k);
    auto t = S * (MatrixDn::Identity(S.cols(), S.cols()) +
                  Slk_inverse.transpose() * S);
    WS.conservativeResize(t.rows(), WS.cols() + t.cols());
    WS.rightCols(t.cols()) = t;
}
auto LI_WS = linearly_independent_system(WS);
```

Вывод программы

Вычисление спектрального радиуса:

```
lambda = (360/7)^(1/5)
```

Вычисление $(\lambda^{-1}C)^*$:

```
w =
```

```
[1, 1/20*(360/7)^(2/5), 1/60*(360/7)^(3/5), 1/4*(360/7)^(1/5), 7/60*(360/7)^(4/5),
1/20*(360/7)^(2/5)]
[7/30*(360/7)^(3/5), 1, 1/5*(360/7)^(1/5), 7/120*(360/7)^(4/5), 7/5*(360/7)^(2/5),
3/5]
[7/6*(360/7)^(2/5), 7/120*(360/7)^(4/5), 1, 7/24*(360/7)^(3/5), 7*(360/7)^(1/5),
7/120*(360/7)^(4/5)]
[7/90*(360/7)^(4/5), 1/5*(360/7)^(1/5), 1/15*(360/7)^(2/5), 1, 7/15*(360/7)^(3/5),
1/5*(360/7)^(1/5)]
[1/6*(360/7)^(1/5), 1/120*(360/7)^(3/5), 1/360*(360/7)^(4/5), 1/24*(360/7)^(2/5), 1,
1/120*(360/7)^(3/5)]
[7/18*(360/7)^(3/5), 1, 1/3*(360/7)^(1/5), 7/72*(360/7)^(4/5), 7/3*(360/7)^(2/5),
1]
```

Вычисление δ:

```
delta = 7*(360/7)^(1/5)
delta^-1 = 1/360*(360/7)^(4/5)
```

Вычисление наихудшего дифференцирующего вектора альтернатив:

```
Linearly independent w1 =
[1/20*(360/7)^(2/5), 1/20*(360/7)^(2/5)]
 [1, 3/5]
 [7/120*(360/7)^(4/5), 7/120*(360/7)^(4/5)]
 [1/5*(360/7)^(1/5), 1/5*(360/7)^(1/5)]
 [1/120*(360/7)^(3/5), 1/120*(360/7)^(3/5)]
 [1, 1]

Worst differentiating weight vector =
[1/60*(360/7)^(3/5)]
 [1/3*(360/7)^(1/5)]
 [1]
 [1/15*(360/7)^(2/5)]
 [1/360*(360/7)^(4/5)]
 [1/3*(360/7)^(1/5)]

D1 = [1, 7/3*(360/7)^(1/5), 3*(360/7)^(1/5)]
 [7, 1, 7/3*(360/7)^(1/5)]
 [9, 7, 1]
nu_1 = (360/7)^(1/10)*sqrt(27)

Worst differentiating vector of alternatives
x1 =
[1/9*(360/7)^(1/10)*sqrt(27)]
 [1]
 [1]
```

Вычисление наилучшего дифференцирующего вектора альтернатив:

```
Computing the best differenting weight vector
```

```
P =
 [1/20*(360/7)^(2/5), 1/20*(360/7)^(2/5)]
 [1, 3/5]
 [7/120*(360/7)^(4/5), 7/120*(360/7)^(4/5)]
 [1/5*(360/7)^(1/5), 1/5*(360/7)^(1/5)]
 [1/120*(360/7)^(3/5), 1/120*(360/7)^(3/5)]
 [1, 1]

Computing w_2

Linearly independent w2 =
 [1/20*(360/7)^(2/5), 1/20*(360/7)^(2/5)]
 [1, 3/5]
 [7/120*(360/7)^(4/5), 7/120*(360/7)^(4/5)]
 [1/5*(360/7)^(1/5), 1/5*(360/7)^(1/5)]
 [1/120*(360/7)^(3/5), 1/120*(360/7)^(3/5)]
 [1, 1]

Computing the best differentiating vector of alternatives
D2 =
 [7/120*(360/7)^(4/5), 7, 9]
 [49/120*(360/7)^(4/5), 7/120*(360/7)^(4/5), 7]
 [21/40*(360/7)^(4/5), 49/120*(360/7)^(4/5), 7/120*(360/7)^(4/5)]
nu_2 = (360/7)^(2/5)*sqrt(189/40)

Best differentiang vector of alternatives
x2 =
 [1/27*(360/7)^(3/5)*sqrt(189/40), 1/27*(360/7)^(3/5)*sqrt(189/40)]
 [1/21*(360/7)^(3/5)*sqrt(189/40), 7/9]
 [1, 1]
```